BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

DNA berulang dimana-mana di dalam genom eukariotik (Charlesworth, Sniegowski, & Stephan, 1994). Identifikasi dan klasifikasi pengulangan adalah tugas anotasi mendasar yang penting karena beberapa alasan. Pertama, pengulangan diyakini memainkan peran penting dalam evolusi genom (Bowen & Joran, 2002) dan penyakit (Buard & Jeffreys, 1997). Kedua, elemen seluler (transposon dan retrotransposons) mungkin berisi wilayah pengkodean yang sulit dibedakan dari tipe gen lainnya. Akhirnya, pengulangan sering menyebabkan banyak keberpihakan lokal, perakitan urutan yang rumit, perbandingan antara genom dan analisis duplikasi skala besar dan penataan ulang (Edgar & Myers, 2005).

Dalam satu dekade terakhir para ilmuwan harus melakukan penelitian laboratorium selama 3 tahun untuk menganalisa DNA (Pahadia, Srivastava, Srivastava, & Patil, 2015). Salah satu kasus dari analisa DNA yang membutuhkan waktu dan tenaga dalam skala besar tersebut adalah untuk menganalisa penyakit yang disebabkan oleh pola genom yang berulang atau disebut dengan *genomic repeats* (Edgar & Myers, 2005) seperti tiga pasang basa berulang yang dapat menyebabkan penyakit dalam kategori *trinucleotide repeat disorders* (Orr & Zoghbi, 2014).

Banyaknya data biologis dari mulai *genome* manusia hingga *genome* hewan ataupun tumbuhan yang dapat kita kumpulkan pada saat ini sangatlah tidak terbatas. Oleh karenanya data yang sangat banyak tersebut menjadi suatu tantangan tersendiri dalam dunia komputerisasi biologi. *Bioinformatics*, merupakan aplikasi dari teknik komputasi untuk menganalisa suatu informasi yang berhubungan dengan molekul biologi dalam skala yang besar, atau yang sekarang biasa disebut dengan molekular biologi, dan meliputi sebagian besar dari struktural biologi, seperti contoh mempelejari tentang genomics (Luscombe, Greenbaum, & Gerstein, 2001).

Dalam menganalisa masalah *genomic repeats* dilakukan analisa *string matching* atau *pattern matching* dimana akan mencari sebuah pola dalam sebuah teks yang berukuran besar. Algoritma dasar untuk pencarian string atau pola ini dengan cara mencocokan segala kemungkinan yang terdapat dalam data dari indeks pertama pada teks hingga selesai. Algoritma ini dikenal dengan *Brute Force (Naïve) Algorithm* yang memiliki kompleksitas dengan kemungkinan terburuknya adalah O(mn) dimana akan sangat memakan waktu yang lama jika semakin banyak teks yang akan dijadikan objek pencarian string atau pola (Al Kindhi & Sardjono, 2015).

Kebutuhan akan pencarian *string* maupun pola dari sebuah data yang seiring perkembangan zaman semakin besar membuat para ilmuwan mengembangkan algoritma yang lebih efisien dari algoritma *brute force* yang mencocokkan satu persatu pola dengan teks yang dapat mengakibatkan besarnya kompleksitas. Oleh karena itu beberapa algoritma *string matching* dikembangkan. Algoritma Knuth-Morris-Pratt merupakan sebuah Algoritma pencarian string yang bekerja dengan memanfaatkan pergeseran pattern teks dari kiri kekanan dalam pencocokkan string dalam teks. Algoritma Knuth- Morris-Pratt dikembangkan pertama kali oleh Donald E. Knuth pada tahun 1967 dan kemudian dilanjutkan oleh James H. Morris bersama Vaughan R. Pratt pada tahun 1966. Kemudian ditahun 1977 Algoritma Knuth-Morris-Pratt (KMP) dipublikasikan (Knuth & Morris, 1977). Algoritma Boyer-Moore (BM) (Boyer & Moore, 1977). Secara teoritis, Algoritma Boyer-Moore adalah salah satu algoritma paling efisien dibandingkan dengan algoritma *string matching* yang lain (Sheik, Aggarwal, Poddar, Balakrishnan, & Sekar, 2004). Algoritma Boyer-Moore memproses pola dan membuat dua tabel, yang dikenal sebagai tabel Boyer-Moore *Bad Character* (bmBc) dan tabel Boyer-Moore *good-suffix* (bmGs). Untuk setiap karakter dalam set alfabet, tabel *bad character* menyimpan nilai pergeseran berdasarkan kemunculan karakter dalam pola. Algoritma ini membentuk dasar untuk beberapa algoritma pencocokan pola. Knuth Morris-Pratt atau / dan Boyer-Moore dapat berfungsi sebagai alat untuk identical urutan identik pada urutan sumber atau cari subsequence berulang (Haponiuk, Pawekowicz, Przybecki, & Nowak, 2017). Algoritma tipe BM (Boyer-Moore) untuk pencocokan pola terkompresi dalam kolase sistem, dan menunjukkan

bahwa sebuah instance dari pencarian algoritma di BPE (*Byte Per Encoding*) dikompresi teks 1.2 ~ 3.0 lebih cepat daripada yang dilakukan *software package agrep* (alat *pattern mathing* tercepat) dalam teks aslinya (Shibata, Matsumoto, & Takeda, 2000). Hasil penelitian tentang analisa perbandingan kecepata Algoritma Boyer Moore dan Algoritma Knuth Morris Pratt (KMP) dengan adanya total nilai dari Metode Perbandingan Eksponesal (MPE) yang dilakukan maka terbukti bahwa Algoritma Boyer Moore merupakan Algoritma yang tercepat dalam melakukan perencarian (Fau, Mesran, & Ginting, 2017).

Dengan munculnya hasil yang tinggi pada data genomik, ilmuwan ataupun para ahli biologi mulai bergulat dengan set data-data besar, bertemu tantangan dengan penanganan, pemrosesan, dan pemindahan informasi yang dulunya dipakai astronom dan fisikawan energi tinggi. Setiap tahun yang berlalu, para ahli biologi mulai merubah dari pengolahan data yang tradisional menjadi lebih ke arah *Big Data* untuk menyelidiki semua permasalahan biologis, dari pengaturan gen dan evolusi genom, mengapa mikroba tinggal di rongga tubuh manusia, dan bagaimana susunan genetik dari berbagai jenis kanker mempengaruhi bagaimana pasien menderita kanker.. Dengan peluang yang belum pernah ada sebelumnya untuk wawasan baru dengan menambang data tekstual yang sangat banyak dan terus bertambah (Pak & Paroubek, 2010).

Para peneliti *Bioinformatics* pada saat ini sedang berhadapan dengan analisis dari suatu data dengan skala yang sangat besar, suatu masalah yang akan populer di masa mendatang. Beberapa pengembangan di perangkat lunak terbuka, yaitu proyek *Hadoop* dan perangkat lunak lainnya yang berhubungan dengan *Hadoop*, melakukan penemuan untuk memberikan skala pada penyimpanan *petabyte* data dalam *Linux*¸ memberikan toleransi kesalahan suatu program apabila terjadi suatu kesalahan analisis data menggunakan *style* pemrograman yang bernama *MapReduce* (Taylor, 2010). Namun, karena pola akses I / O berbasis Hadoop, hasil perhitungan menengah tidak di-*cache*. Oleh karena itu, Hadoop hanya cocok untuk pemrosesan data batch, dan menunjukkan kinerja yang buruk untuk pemrosesan data berulang (Guo et al., 2018). Untuk mengatasi masalah tersebut, maka ditemukanlah Apache spark, kerangka kerja komputasi yang

lebih cepat dan dirancang khusus untuk menangani sejumlah data besar (Zaharia et al., 2010).

Tidak seperti komputasi berbasis disk layaknya Hadoop, Spark melakukan komputasi memori dengan memperkenalkan *Resilient Distributed Dataset* (RDD) yang tangguh. Karena dimungkinkan untuk menyimpan hasil dalam memori, hal tersebut lebih efisien untuk operasi berulang. Dalam hal kinerja, Spark dapat mencapai 100 kali lebih cepat dalam hal akses memori daripada Hadoop (Zaharia et al., 2010). Kesenjangan antara Spark dan Hadoop lebih dari 10 kali lipat lebih besar, bahkan jika kita membandingkan di antara keduanya berdasarkan kinerja disk (Han & Sql, 2016). Dalam hal fleksibilitas, Spark menyediakan antarmuka pemrograman aplikasi tingkat tinggi (API) di Jawa, Scala, Python, dan R, dan shell interaktif. Dalam hal umum, Spark menyediakan pemrosesan data terstruktur, pembelajaran mesin, komputasi grafik, dan kemampuan komputasi aliran dengan mendukung beberapa komponen canggih (Guo et al., 2018).

Berbeda dengan pemrosesan data volume besar berbasis *batch*, pemrosesan *streaming* mengambil langkah lebih maju terhadap data *streaming*. Dengan pertumbuhan eksponensial dalam data *streaming* yang berkelanjutan, pemrosesan *streaming* telah mendapatkan banyak popularitas. Spark Streaming adalah salah satu *open source framework* untuk pemrosesan *streaming* yang andal, *high-throughput*, dan pemrosesan *streaming* latensi rendah (Liao et al., 2015).

1.2 Rumusan Masalah

Dari latar belakang masalah yang telah dipaparkan, maka dapat diambil rumusan masalah sebagai berikut :

1. Bagaimana mengimplementasikan *string mathing* algoritma Boyer-Moore dalam mendeteksi *genomic repeats*?
2. Bagaimana membuat suatu komputasi *Big Data* dengan menerapkan algoritma Boyer-Moore?
3. Bagaimana mengimplementasikan algoritma Boyer-Moore pada Apache Spark Streaming?

4. Bagaimana hasil dari perbandingan beberapa *cluster* yang digunakan pada komputasi *Big Data* dengan perbedaan yang terdapat pada *core* dan *node* yang dipakai?

1.3 Tujuan Penelitian

Maka tujuan dari penelitian ini adalah :

1. Merancang model komputasi *Big Data* untuk pencarian *pattern* pada *string* menggunakan algoritma Boyer-Moore.
2. Implementasi model komputasi *Big Data* menggunakan algoritma Boyer-Moore pada Apache Spark Streaming dengan bahasa pemrograman Python.
3. Melakukan analisa terkait kecepatan dari hasil eksperimen dengan mengacu pada *core* dan *node* yang dipakai.

1.4 Manfaat Penelitian

Manfaat dari penelitian ini antara lain sebagai berikut :

1. Membuat model komputasi yang dapat memudahkan dunia kedokteran ataupun para peneliti dalam bidang biologi dalam menganalisis *genomic repeats* manusia.
2. Membuat suatu program dan model komputasi *Big Data* yang dapat dikembangkan oleh para peneliti selanjutnya.

1.5 Batasan Masalah

Batasan masalah dari penelitian ini antara lain sebagai berikut :

1. Program ini dapat dijalankan menggunakan masukan data dengan format NCBI/Ensmbl.
2. Data yang digunakan pada penelitian ini merupakan contoh rangkaian DNA manusia dari publikasi nomor 95 pada halaman FTP Ensembl yang dapat diunduh di ftp://ftp.ensembl.org/pub/release-95/fasta/homo_sapiens/dna/
3. Jumlah *core* prosesor pada *cluster* yang digunakan dalam penelitian ini adalah 2, 4, 8, dan 16 *cores.*

4. Jumlah *node* pada *cluster* yang digunakan dalam penelitian ini adalah 2, 4, 5, dan 11 *nodes.*
5. *Node* dan *cluster* yang digunakan pada penelitian ini merupakan *node* dan *cluster* dari Google Cloud Project dengan saldo sebesar $300.

1.6 Sistematika Penulisan

Sistematika dari penelitian ini akan diuraikan terkait penjelasan pada tiap bab.

BAB I PENDAHULUAN

Bab ini menjelaskan mengapa dan bagaimana penelitian akan dilakukan. Diawali dengan latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini menjelaskan tentang teori pendamping atau pendukung untuk melakukan penelitian. Teori yang dijelaskan dalam bab ini yaitu mengenai *genomic repeats*, algoritma Boyer-Moore, *Big Data*, Apache Spark Streaming, serta Google Cloud Project.

BAB III METODOLOGI PENELITIAN

Bab ini menjelaskan langkah-langkah penelitian yang akan dilakukan dimulai dari desain penelitian, fokus penelitian, alat dan bahan yang digunakan untuk penelitian dan yang terkahir adalah metode penelitian.

BAB IV HASIL DAN PEMBAHASAN

Bab ini menjabarkan hasil penelitian dan eksperimen yang telah dilakukan. Semua pertanyaan mengenai masalah yang diangkat dalam tema skripsi dibahas pada bab ini. Beberapa hal di antaranya adalah tentang proses pengumpulan data, pengembangan model, implementasi sistem, studi kasus, desain eksperimen, dan analisa.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran bagi peneliti selanjutnya dari hasil penelitian yang telah dilakukan.